**“Covid-19 Outbreak Prediction Using Machine Learning”**

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **Sr.no.** | **Topics** | **Page no.** |
| **1.** | **Introduction** | 1 |
| 1.1 | -*Present System* | 1 |
| 1.2 | *-Proposed System* | 2 |
| **2.** | **System Design** | 3 |
| 2.1 | *-System flowchart* | 3 |
| 2.2 | *-Dataset* | 4 |
| **3.** | **Hardware and Software details** | 5 |
|  | *.* |  |
| **4.** | **Implementation Work Details** | 6 |
| 4.1 | *-Real life applications* | 6 |
| 4.2 | *-Data implementation and program execution* | 6 |
| **5.** | **Source Code** | 7 |
| **6.** | **Input/output Screens/ Model’s Photograph** | 23 |
| **7.** | **System Testing** | 27 |
| **8.** | **Conclusion** | 28 |
| 8.1 | *-Limitations* | 28 |
| 8.2 | *-Scope for future work* | 28 |
| **9.** | **Bibliography** | 29 |
| **10.** | **Annexures**  - *Plagiarism Report* | 30 |

**List of Figures**

|  |  |  |
| --- | --- | --- |
| **Fig no.** | **Title** | **Page**  **no.** |
| 6.1 | Growth of different types of cases in India | 23 |
| 6.2 | Confirmed cases Linear Regression Prediction | 23 |
| 6.3 | Polynomial Regression Prediction for confirmed cases | 24 |
| 6.4 | SVM regressor Prediction for confirmed cases | 24 |
| 6.5 | Holts Linear Model Prediction for confirmed cases | 25 |
| 6.6 | Holt’s Winter model prediction for confirmed cases | 25 |
| 6.7 | AR model prediction for confirmed cases | 26 |
| 6.8 | SARIMA model Prediction for confirmed cases | 26 |

# Source Code

import warnings warnings.filterwarnings('ignore') import pandas as pd

import matplotlib.pyplot as plt import seaborn as sns

import plotly.express as px import plotly.graph\_objects as go

from plotly.subplots import make\_subplots import numpy as np

import datetime as dt

from datetime import timedelta

from sklearn.model\_selection import GridSearchCV from sklearn.preprocessing import StandardScaler

from sklearn.metrics import silhouette\_score,silhouette\_samples from sklearn.linear\_model import LinearRegression,Ridge,Lasso from sklearn.svm import SVR

from sklearn.metrics import mean\_squared\_error,r2\_score import statsmodels.api as sm

from statsmodels.tsa.api import Holt,SimpleExpSmoothing,ExponentialSmoothing from sklearn.preprocessing import PolynomialFeatures

from statsmodels.tsa.stattools import adfuller from pmdarima import auto\_arima std=StandardScaler()

covid=pd.read\_csv(r"C:\Users\Ritika\Desktop\covid\_19\_data.csv") covid.head()

print("Shape of the dataset: ",covid.shape) print("Checking for null values:\n",covid.isnull().sum())

print("Checking Data-type of each column:\n",covid.dtypes)

#Dropping column as SNo is of no use, and "Province/State" contains too many missing values

covid.drop(["SNo"],1,inplace=True)

#Converting "Observation Date" into Datetime format covid["ObservationDate"]=pd.to\_datetime(covid["ObservationDate"])

grouped\_country=covid.groupby(["Country/Region","ObservationDate"]).agg({"Confirmed ":'sum',"Recovered":'sum',"Deaths":'sum'})

grouped\_country["Active Cases"]=grouped\_country["Confirmed"]- grouped\_country["Recovered"]-grouped\_country["Deaths"]

grouped\_country["log\_confirmed"]=np.log(grouped\_country["Confirmed"]) grouped\_country["log\_active"]=np.log(grouped\_country["Active Cases"])

#Grouping different types of cases as per the date

datewise=covid.groupby(["ObservationDate"]).agg({"Confirmed":'sum',"Recovered":'sum', "Deaths":'sum'})

datewise["Days Since"]=datewise.index-datewise.index.min()

print("Basic Information")

print("Totol number of countries with Disease Spread: ",len(covid["Country/Region"].unique()))

print("Total number of Confirmed Cases around the World: ",datewise["Confirmed"].iloc[- 1])

print("Total number of Recovered Cases around the World: ",datewise["Recovered"].iloc[- 1])

print("Total number of Deaths Cases around the World: ",datewise["Deaths"].iloc[-1])

print("Total number of Active Cases around the World: ",(datewise["Confirmed"].iloc[-1]- datewise["Recovered"].iloc[-1]-datewise["Deaths"].iloc[-1]))

print("Total number of Closed Cases around the World: ",datewise["Recovered"].iloc[- 1]+datewise["Deaths"].iloc[-1])

print("Number of Confirmed Cases in last 24 hours: ",datewise["Confirmed"].iloc[-1]- datewise["Confirmed"].iloc[-2])

print("Number of Recovered Cases in last 24 hours: ",datewise["Recovered"].iloc[-1]- datewise["Recovered"].iloc[-2])

print("Number of Death Cases in last 24 hours: ",datewise["Deaths"].iloc[-1]- datewise["Deaths"].iloc[-2])

fig=px.bar(x=datewise.index,y=datewise["Confirmed"]-datewise["Recovered"]- datewise["Deaths"])

fig.update\_layout(title="Distribution of Number of Active Cases", xaxis\_title="Date",yaxis\_title="Number of Cases",)

fig.show()

india\_data=covid[covid["Country/Region"]=="India"]

datewise\_india=india\_data.groupby(["ObservationDate"]).agg({"Confirmed":'sum',"Recove red":'sum',"Deaths":'sum'})

print(datewise\_india.iloc[-1])

print("Total Active Cases: ",datewise\_india["Confirmed"].iloc[-1]- datewise\_india["Recovered"].iloc[-1]-datewise\_india["Deaths"].iloc[-1])

print("Total Closed Cases: ",datewise\_india["Recovered"].iloc[- 1]+datewise\_india["Deaths"].iloc[-1])

fig=go.Figure()

fig.add\_trace(go.Scatter(x=datewise\_india.index, y=datewise\_india["Confirmed"], mode='lines+markers',

name='Confirmed Cases')) fig.add\_trace(go.Scatter(x=datewise\_india.index, y=datewise\_india["Recovered"],

mode='lines+markers', name='Recovered Cases'))

fig.add\_trace(go.Scatter(x=datewise\_india.index, y=datewise\_india["Deaths"], mode='lines+markers',

name='Death Cases'))

fig.update\_layout(title="Growth of different types of cases in India", xaxis\_title="Date",yaxis\_title="Number of

Cases",legend=dict(x=0,y=1,traceorder="normal")) fig.show()

datewise["Days Since"]=datewise.index-datewise.index[0] datewise["Days Since"]=datewise["Days Since"].dt.days

train\_ml=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid\_ml=datewise.iloc[int(datewise.shape[0]\*0.95):] model\_scores=[]

lin\_reg=LinearRegression(normalize=True)

lin\_reg.fit(np.array(train\_ml["Days Since"]).reshape(- 1,1),np.array(train\_ml["Confirmed"]).reshape(-1,1))

prediction\_valid\_linreg=lin\_reg.predict(np.array(valid\_ml["Days Since"]).reshape(-1,1))

model\_scores.append(np.sqrt(mean\_squared\_error(valid\_ml["Confirmed"],prediction\_valid

\_linreg)))

print("Root Mean Square Error for Linear Regression: ",np.sqrt(mean\_squared\_error(valid\_ml["Confirmed"],prediction\_valid\_linreg)))

plt.figure(figsize=(11,6)) prediction\_linreg=lin\_reg.predict(np.array(datewise["Days Since"]).reshape(-1,1)) linreg\_output=[]

for i in range(prediction\_linreg.shape[0]): linreg\_output.append(prediction\_linreg[i][0])

fig=go.Figure()

fig.add\_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=datewise.index, y=linreg\_output, mode='lines',name="Linear Regression Best Fit Line", line=dict(color='black', dash='dot')))

fig.update\_layout(title="Confirmed Cases Linear Regression Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

train\_ml=datewise.iloc[:int(datewise.shape[0]\*0.95)]

valid\_ml=datewise.iloc[int(datewise.shape[0]\*0.95):]

poly = PolynomialFeatures(degree = 8)

train\_poly=poly.fit\_transform(np.array(train\_ml["Days Since"]).reshape(-1,1)) valid\_poly=poly.fit\_transform(np.array(valid\_ml["Days Since"]).reshape(-1,1)) y=train\_ml["Confirmed"]

linreg=LinearRegression(normalize=True) linreg.fit(train\_poly,y)

prediction\_poly=linreg.predict(valid\_poly) rmse\_poly=np.sqrt(mean\_squared\_error(valid\_ml["Confirmed"],prediction\_poly)) model\_scores.append(rmse\_poly)

print("Root Mean Squared Error for Polynomial Regression: ",rmse\_poly)

comp\_data=poly.fit\_transform(np.array(datewise["Days Since"]).reshape(-1,1)) plt.figure(figsize=(11,6))

predictions\_poly=linreg.predict(comp\_data)

fig=go.Figure()

fig.add\_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=datewise.index, y=predictions\_poly, mode='lines',name="Polynomial Regression Best Fit",

line=dict(color='black', dash='dot'))) fig.update\_layout(title="Confirmed Cases Polynomial Regression Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases", legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

new\_prediction\_poly=[] for i in range(1,18):

new\_date\_poly=poly.fit\_transform(np.array(datewise["Days Since"].max()+i).reshape(- 1,1))

new\_prediction\_poly.append(linreg.predict(new\_date\_poly)[0])

train\_ml=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid\_ml=datewise.iloc[int(datewise.shape[0]\*0.95):]

#Intializing SVR Model svm=SVR(C=1,degree=6,kernel='poly',epsilon=0.01)

#Fitting model on the training data

svm.fit(np.array(train\_ml["Days Since"]).reshape(- 1,1),np.array(train\_ml["Confirmed"]).reshape(-1,1))

prediction\_valid\_svm=svm.predict(np.array(valid\_ml["Days Since"]).reshape(-1,1))

model\_scores.append(np.sqrt(mean\_squared\_error(valid\_ml["Confirmed"],prediction\_valid

\_svm)))

print("Root Mean Square Error for Support Vectore Machine: ",np.sqrt(mean\_squared\_error(valid\_ml["Confirmed"],prediction\_valid\_svm)))

plt.figure(figsize=(11,6)) prediction\_svm=svm.predict(np.array(datewise["Days Since"]).reshape(-1,1)) fig=go.Figure()

fig.add\_trace(go.Scatter(x=datewise.index, y=datewise["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=datewise.index, y=prediction\_svm, mode='lines',name="Support Vector Machine Best fit Kernel", line=dict(color='black', dash='dot')))

fig.update\_layout(title="Confirmed Cases Support Vectore Machine Regressor Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

new\_date=[] new\_prediction\_lr=[] new\_prediction\_svm=[] for i in range(1,18):

new\_date.append(datewise.index[-1]+timedelta(days=i))

new\_prediction\_lr.append(lin\_reg.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0][0])

new\_prediction\_svm.append(svm.predict(np.array(datewise["Days Since"].max()+i).reshape(-1,1))[0])

pd.set\_option('display.float\_format', lambda x: '%.6f' % x)

model\_predictions=pd.DataFrame(zip(new\_date,new\_prediction\_lr,new\_prediction\_poly,ne w\_prediction\_svm),

columns=["Dates","Linear Regression Prediction","Polynonmial Regression Prediction","SVM Prediction"])

model\_predictions.head()

model\_train=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid=datewise.iloc[int(datewise.shape[0]\*0.95):] y\_pred=valid.copy()

holt=Holt(np.asarray(model\_train["Confirmed"])).fit(smoothing\_level=0.4, smoothing\_slope=0.4,optimized=False)

y\_pred["Holt"]=holt.forecast(len(valid))

model\_scores.append(np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["Holt"])))

print("Root Mean Square Error Holt's Linear Model: ",np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["Holt"])))

fig=go.Figure()

fig.add\_trace(go.Scatter(x=model\_train.index, y=model\_train["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=valid.index, y=valid["Confirmed"], mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add\_trace(go.Scatter(x=valid.index, y=y\_pred["Holt"], mode='lines+markers',name="Prediction of Confirmed Cases",))

fig.update\_layout(title="Confirmed Cases Holt's Linear Model Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

holt\_new\_date=[] holt\_new\_prediction=[] for i in range(1,18):

holt\_new\_date.append(datewise.index[-1]+timedelta(days=i)) holt\_new\_prediction.append(holt.forecast((len(valid)+i))[-1])

model\_predictions["Holt's Linear Model Prediction"]=holt\_new\_prediction model\_predictions.head()

model\_train=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid=datewise.iloc[int(datewise.shape[0]\*0.95):] y\_pred=valid.copy()

es=ExponentialSmoothing(np.asarray(model\_train['Confirmed']),seasonal\_periods=14,trend

='add', seasonal='mul').fit()

y\_pred["Holt's Winter Model"]=es.forecast(len(valid))

model\_scores.append(np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["Holt's Winter Model"])))

print("Root Mean Square Error for Holt's Winter Model: ",np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["Holt's Winter Model"])))

fig=go.Figure()

fig.add\_trace(go.Scatter(x=model\_train.index, y=model\_train["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=valid.index, y=valid["Confirmed"], mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add\_trace(go.Scatter(x=valid.index, y=y\_pred["Holt\'s Winter Model"], mode='lines+markers',name="Prediction of Confirmed Cases",))

fig.update\_layout(title="Confirmed Cases Holt's Winter Model Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

holt\_winter\_new\_prediction=[] for i in range(1,18):

holt\_winter\_new\_prediction.append(es.forecast((len(valid)+i))[-1]) model\_predictions["Holt's Winter Model Prediction"]=holt\_winter\_new\_prediction model\_predictions.head()

model\_train=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid=datewise.iloc[int(datewise.shape[0]\*0.95):] y\_pred=valid.copy()

model\_ar= auto\_arima(model\_train["Confirmed"],trace=True, error\_action='ignore', start\_p=0,start\_q=0,max\_p=4,max\_q=0,

suppress\_warnings=True,stepwise=False,seasonal=False) model\_ar.fit(model\_train["Confirmed"])

prediction\_ar=model\_ar.predict(len(valid)) y\_pred["AR Model Prediction"]=prediction\_ar

model\_scores.append(np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["AR Model Prediction"])))

print("Root Mean Square Error for AR Model: ",np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["AR Model Prediction"])))

fig=go.Figure()

fig.add\_trace(go.Scatter(x=model\_train.index, y=model\_train["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=valid.index, y=valid["Confirmed"], mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add\_trace(go.Scatter(x=valid.index, y=y\_pred["AR Model Prediction"], mode='lines+markers',name="Prediction of Confirmed Cases",))

fig.update\_layout(title="Confirmed Cases AR Model Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

AR\_model\_new\_prediction=[] for i in range(1,18):

AR\_model\_new\_prediction.append(model\_ar.predict(len(valid)+i)[-1]) model\_predictions["AR Model Prediction"]=AR\_model\_new\_prediction model\_predictions.head()

model\_train=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid=datewise.iloc[int(datewise.shape[0]\*0.95):] y\_pred=valid.copy()

model\_sarima= auto\_arima(model\_train["Confirmed"],trace=True, error\_action='ignore', start\_p=0,start\_q=0,max\_p=2,max\_q=2,m=7,

suppress\_warnings=True,stepwise=True,seasonal=True) model\_sarima.fit(model\_train["Confirmed"])

model\_ma= auto\_arima(model\_train["Confirmed"],trace=True, error\_action='ignore', start\_p=0,start\_q=0,max\_p=0,max\_q=2,

suppress\_warnings=True,stepwise=False,seasonal=False) model\_ma.fit(model\_train["Confirmed"])

prediction\_ma=model\_ma.predict(len(valid)) y\_pred["MA Model Prediction"]=prediction\_ma

model\_scores.append(np.sqrt(mean\_squared\_error(valid["Confirmed"],prediction\_ma)))

print("Root Mean Square Error for MA Model: ",np.sqrt(mean\_squared\_error(valid["Confirmed"],prediction\_ma)))

fig=go.Figure()

fig.add\_trace(go.Scatter(x=model\_train.index, y=model\_train["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=valid.index, y=valid["Confirmed"], mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add\_trace(go.Scatter(x=valid.index, y=y\_pred["MA Model Prediction"], mode='lines+markers',name="Prediction for Confirmed Cases",))

fig.update\_layout(title="Confirmed Cases MA Model Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

MA\_model\_new\_prediction=[]

for i in range(1,18): MA\_model\_new\_prediction.append(model\_ma.predict(len(valid)+i)[-1])

model\_predictions["MA Model Prediction"]=MA\_model\_new\_prediction model\_predictions.head()

model\_train=datewise.iloc[:int(datewise.shape[0]\*0.95)] valid=datewise.iloc[int(datewise.shape[0]\*0.95):] y\_pred=valid.copy()

model\_arima= auto\_arima(model\_train["Confirmed"],trace=True, error\_action='ignore', start\_p=1,start\_q=1,max\_p=3,max\_q=3,

suppress\_warnings=True,stepwise=False,seasonal=False) model\_arima.fit(model\_train["Confirmed"])

prediction\_arima=model\_arima.predict(len(valid)) y\_pred["ARIMA Model Prediction"]=prediction\_arima

model\_scores.append(np.sqrt(mean\_squared\_error(valid["Confirmed"],prediction\_arima)))

print("Root Mean Square Error for ARIMA Model: ",np.sqrt(mean\_squared\_error(valid["Confirmed"],prediction\_arima)))

fig=go.Figure()

fig.add\_trace(go.Scatter(x=model\_train.index, y=model\_train["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=valid.index, y=valid["Confirmed"], mode='lines+markers',name="Validation Data for Confirmed Cases",))

fig.add\_trace(go.Scatter(x=valid.index, y=y\_pred["ARIMA Model Prediction"],

mode='lines+markers',name="Prediction for Confirmed Cases",)) fig.update\_layout(title="Confirmed Cases ARIMA Model Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

ARIMA\_model\_new\_prediction=[] for i in range(1,18):

ARIMA\_model\_new\_prediction.append(model\_arima.predict(len(valid)+i)[-1]) model\_predictions["ARIMA Model Prediction"]=ARIMA\_model\_new\_prediction model\_predictions.head()

prediction\_sarima=model\_sarima.predict(len(valid)) y\_pred["SARIMA Model Prediction"]=prediction\_sarima

model\_scores.append(np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["SARIMA Model Prediction"])))

print("Root Mean Square Error for SARIMA Model: ",np.sqrt(mean\_squared\_error(y\_pred["Confirmed"],y\_pred["SARIMA Model Prediction"])))

fig=go.Figure()

fig.add\_trace(go.Scatter(x=model\_train.index, y=model\_train["Confirmed"], mode='lines+markers',name="Train Data for Confirmed Cases"))

fig.add\_trace(go.Scatter(x=valid.index, y=valid["Confirmed"],

mode='lines+markers',name="Validation Data for Confirmed Cases",)) fig.add\_trace(go.Scatter(x=valid.index, y=y\_pred["SARIMA Model Prediction"],

mode='lines+markers',name="Prediction for Confirmed Cases",)) fig.update\_layout(title="Confirmed Cases SARIMA Model Prediction",

xaxis\_title="Date",yaxis\_title="Confirmed Cases",legend=dict(x=0,y=1,traceorder="normal"))

fig.show()

SARIMA\_model\_new\_prediction=[] for i in range(1,18):

SARIMA\_model\_new\_prediction.append(model\_sarima.predict(len(valid)+i)[-1]) model\_predictions["SARIMA Model Prediction"]=SARIMA\_model\_new\_prediction model\_predictions.head()

model\_names=["Linear Regression","Polynomial Regression","Support Vector Machine Regressor","Holt's Linear","Holt's Winter Model",

"Auto Regressive Model (AR)","Moving Average Model (MA)","ARIMA Model","SARIMA Model"]

model\_summary=pd.DataFrame(zip(model\_names,model\_scores),columns=["Model Name","Root Mean Squared Error"]).sort\_values(["Root Mean Squared Error"])

model\_summary

# Chapter 6: Output Screens

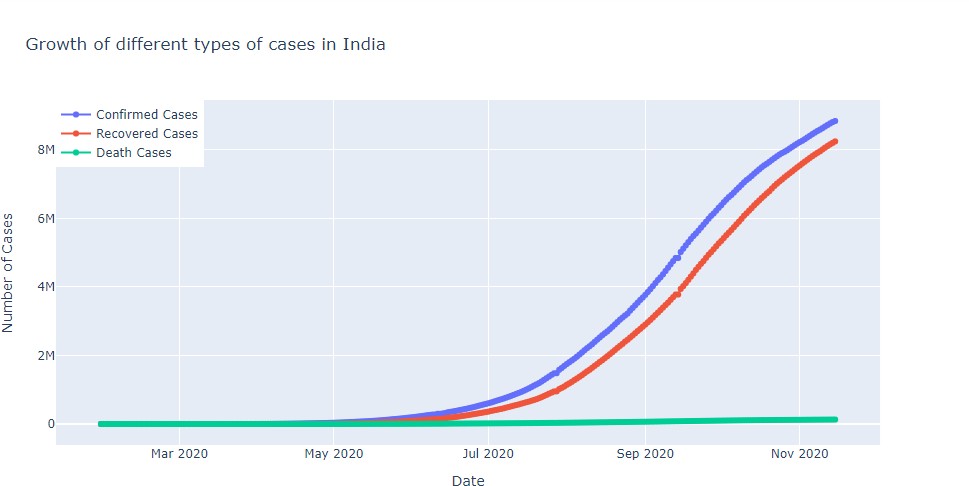


Fig 6.1.Growth of different types of cases in India

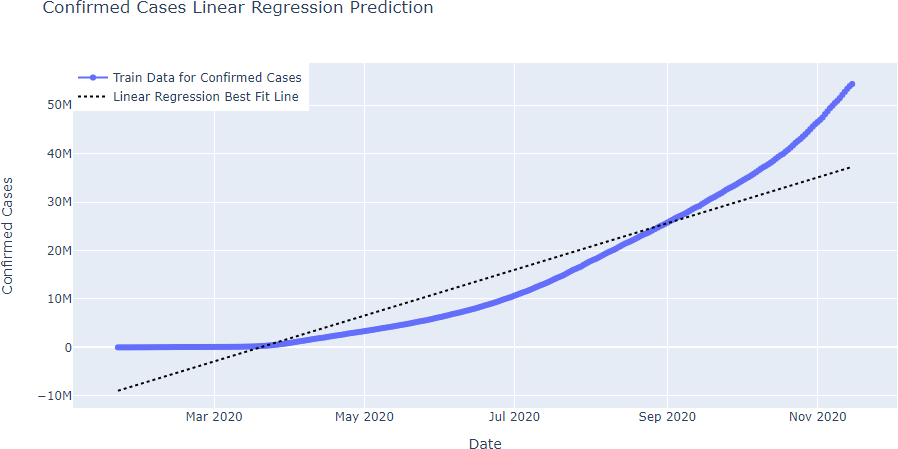


Fig 6.2 Confirmed cases Linear Regression Prediction

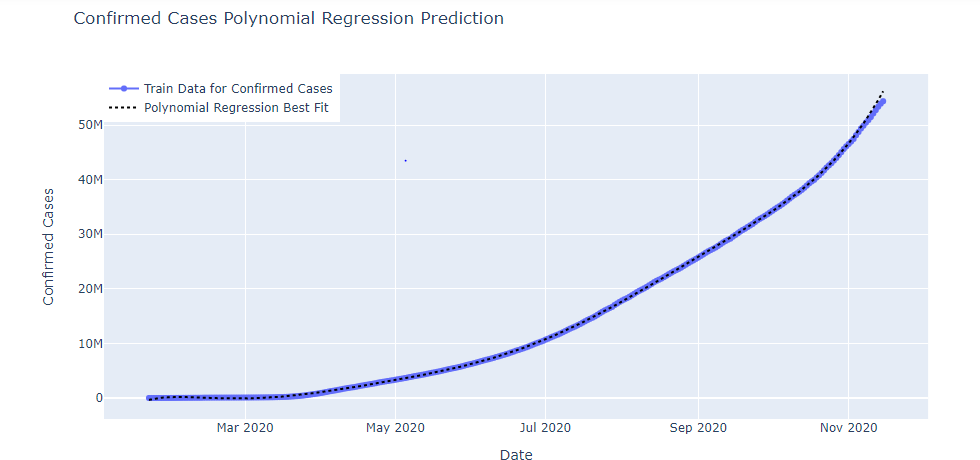


Fig 6.3. Polynomial Regression Prediction for confirmed cases

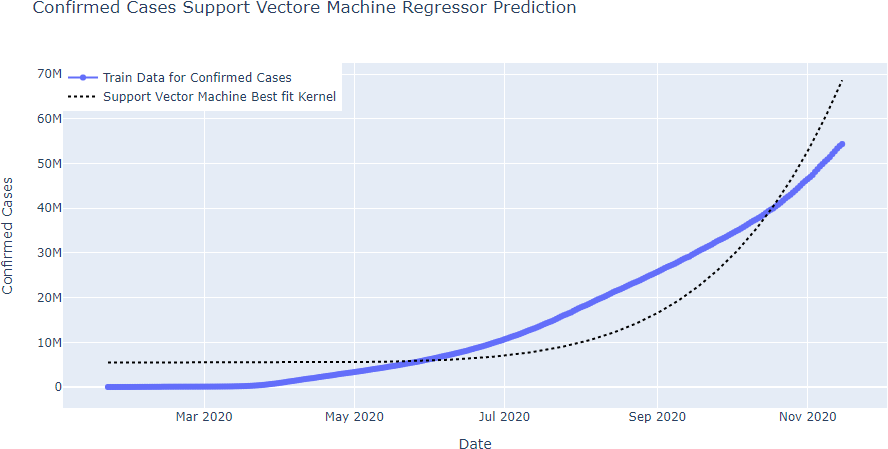


Fig 6.4. SVM regressor Prediction for confirmed cases

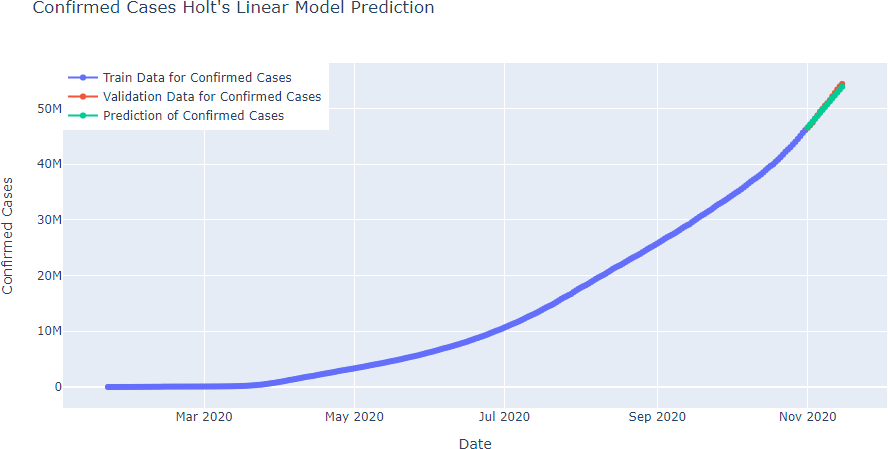


Fig 6.5 Holts Linear Model Prediction for confirmed cases

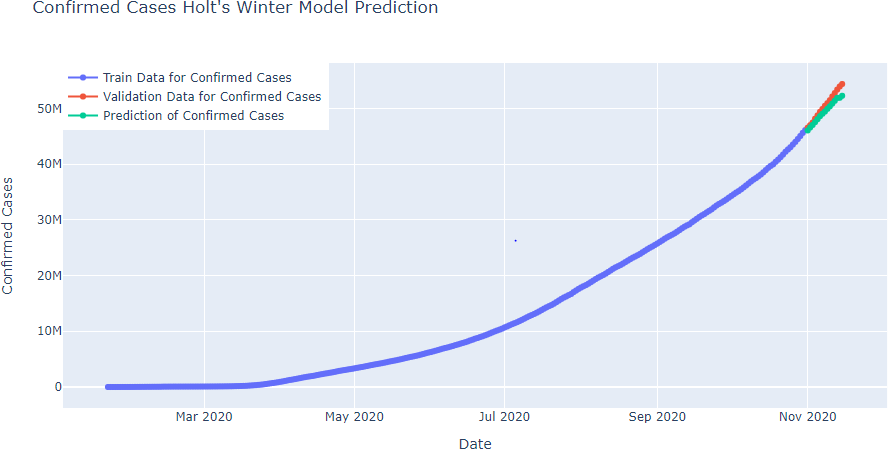


Fig 6.6. Holt’s Winter model prediction for confirmed cases

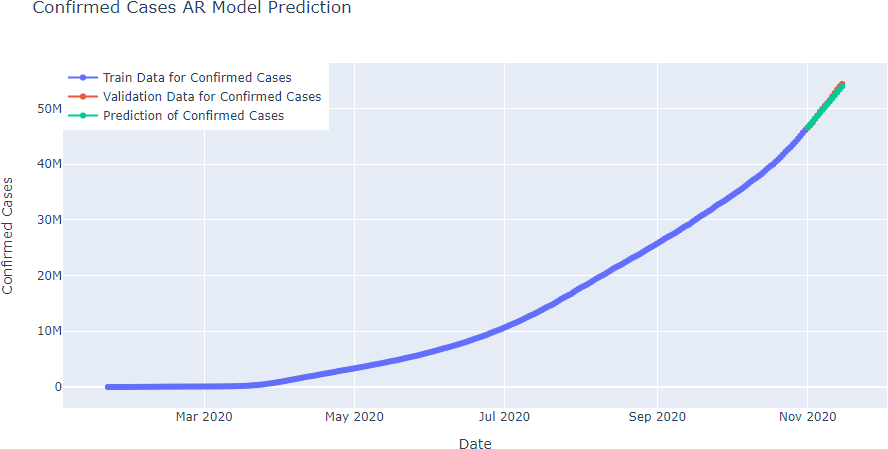


Fig 6.7. AR model prediction for confirmed cases

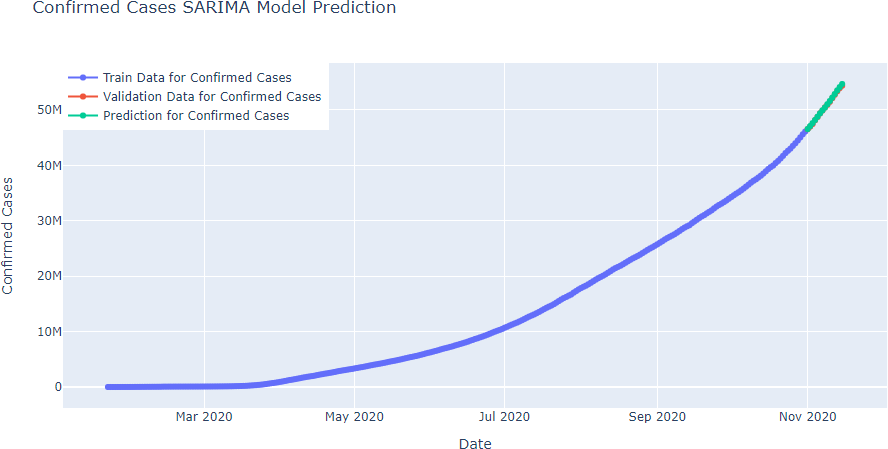


Fig 6.8. SARIMA model Prediction

## Bibliography

1.G. Bontempi, S. B. Taieb and Y.-A. Le Borgne, "Machine learning strategies for time series forecasting", *Proc. Eur. Bus. Intell. Summer School*, pp. 62-77, 2012.

1. F. Petropoulos and S. Makridakis, "Forecasting the novel coronavirus COVID-19", *PLoS ONE*, vol. 15, no. 3, Mar. 2020.
2. G. Grasselli, A. Pesenti and M. Cecconi, "Critical care utilization for the COVID-19 outbreak in lombardy italy: Early experience and forecast during an emergency response", *JAMA*, vol. 323, no. 16, pp. 1545, Apr. 2020.
3. S. Makridakis, E. Spiliotis and V. Assimakopoulos, "Statistical and machine learning forecasting methods: Concerns and ways forward", *PLoS ONE*, vol. 13, Mar. 2018.